TMO Progress Report 42-142 August 15, 2000

Stopping Rules for Turbo Decoders

A. Matache,[1] S. Dolinar,[1] and F. Pollara[1]

Decoders for turbo codes are iterative in nature, i.e., they have to perform a certain number of iterations before reaching a satisfactory degree of confidence regarding a frame to be decoded. Until now standard turbo decoders have used a fixed number of iterations. In this article, we propose some simple "stopping rules" that can be used to reduce the average number of iterations. This technique offers a trade-off between speed and performance and can provide a significant increase in the average decoding speed while not sacrificing decoder performance.

We tested several types of stopping rules for turbo decoders. One type is based on comparing decoded bits (hard bit decisions) with previous decoded bits; a second type is based on comparing reliabilities (soft bit decisions) with a threshold; and a third type uses a cyclic redundancy check (CRC) code applied to hard decoded bits. We simulated turbo decoder performance using these rules (including several variations of the first two types) and further required that the decoder cease after 20 iterations if the stopping rule is not yet satisfied. Specifically, we analyzed the decoder-error rates and the average number of iterations for each rule.

We found that the average number of iterations was roughly between 4 and 7 for a bit signal-to-noise ratio, E_b/N_0, near the "waterfall" threshold, as compared with the 10 fixed iterations used by the current turbo decoder. In addition, the resulting error rates were noticeably lower than those for 10 fixed iterations, and in fact were very nearly equal to the error rates achieved by a decoder using 20 fixed iterations.

I. Introduction

Turbo codes [1] achieve exceptional, near-Shannon-limit error-correction performance with low decoding complexity. A turbo encoder consists of two simple recursive convolutional encoders, separated by a K-bit interleaver. The turbo decoder consists of two decoders individually matched to the simple constituent codes. Each decoder sends likelihood estimates of the decoded bits to the other decoder and uses the corresponding estimates from the other decoder as a priori likelihoods. The overall turbo decoder iterates between the outputs of the two constituent decoders until reaching satisfactory convergence.

Traditionally, the turbo decoding algorithm uses a fixed number of iterations, N, per frame. For example, $N = 10$ is a typical fixed number of iterations used for decoding the family of turbo codes

---

[1] Communications Systems and Research Section.

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

proposed for the new Consultative Committee for Space Data Systems (CCSDS) standard [7]. Stopping after a fixed number of iterations, N, is the simplest stopping rule to compute, requiring nothing more than a simple counter that is incremented and checked independently of any knowledge available during the decoding process. However, selecting an appropriate value of N requires a trade-off between decoding performance and speed. Figure 1 shows a typical variation of codeword-error rate, or frame-error rate (FER), with the number of iterations, for the rate-1/3 code with a block size of 1784 from the family of turbo codes proposed for the CCSDS, operating at a bit signal-to-noise ratio (bit SNR) of $E_b/N_0 = 0.6$ dB.

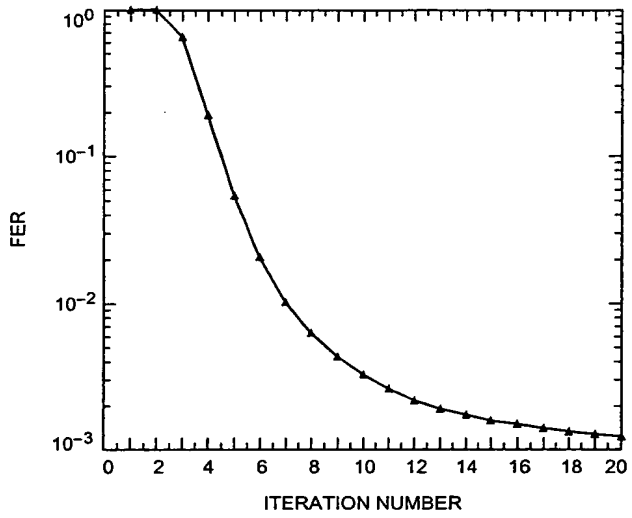

Fig. 1. Variation of FER with the number of iterations.

We see that an FER of about 3×10^{-3} is achieved at this E_b/N_0 with 10 iterations, and further iterations do not reduce this error rate dramatically, reaching an FER of about 1×10^{-3} at 20 iterations. Conversely, significant performance penalties do result from reducing the number of iterations below 10. For example, the FER rises more than tenfold to 5×10^{-2} if only 5 iterations are used, and thus an $N = 5$ fixed stopping condition would not produce a very reasonable trade-off of performance versus decoding speed for this code. On the other hand, after the end of the fifth iteration, the decoder is wasting effort by continuing to iterate on the 95 percent of frames that already are decodable by then.

It is possible to improve the average decoding speed of the turbo decoder if a stopping rule with a variable number of iterations per frame is used instead [2–6]. For each decoded frame, the number of iterations performed is determined by the number of passes before a certain condition or rule for stopping is satisfied. The stopping condition attempts to determine when a frame can be reliably decoded with no further iterations, and it is computed based on data available to the decoder during the decoding of each specific codeword. More explicitly, at the end of each iteration, the decoder performs a check on the condition for stopping. If the condition is true, the iterative process on the frame is terminated, and the decoded sequence from the current iteration is sent to the output; otherwise, the iterative process continues to the next iteration. To prevent an endless loop should the stopping rule never be satisfied, we require that the decoder cease after a maximum number of iterations, $N_{\max}$.

In the next section, we define several stopping rules that cause the decoder to use a variable number of iterations. These rules are divided into two main types, hard-decision rules and soft-decision rules. The hard and soft stopping rules proposed here are ad hoc rules that are easily computable from data available during the normal decoding operation. This is important, because the value of a stopping rule

can be offset by the computational complexity required to test the stopping condition. We also define two additional stopping rules that rely on side information, one realizable using an error-detecting code and one unrealizable but useful as a performance benchmark.

In Section III, we investigate the trade-off between the decoder's error-rate performance and the average number of iterations, briefly mention some additional computational complexity considerations, and discuss the need for increased buffering of data to accommodate the variability in the number of iterations required to decode each frame.

II. Stopping Rule Definitions

Hard-decision rules attempt to detect unreliable decoded sequences by evaluating the tentative decoded bits (hard bit decisions) at the end of each iteration or half-iteration. Soft-decision rules are based on comparing a metric on bit reliabilities (soft bit decisions) with a threshold. A CRC rule detects unreliable decoded sequences using an outer cyclic redundancy check (CRC) code applied to hard decoded bits. A "magic genie" rule is like the CRC rule, except it uses a mythical error-detecting code that never fails to detect an error.

A. Hard-Decision Rules

The hard-decision rules H_1 through H_4 simply check whether identical tentative bit decisions are made at successive iterations or half-iterations.

1. Hard Rule H_1 (consistency in two successive half-iterations). With this rule, after each iteration both component decoders make tentative decoded bit decisions, and the iterative process is stopped at the earliest iteration, $n \leq N_{\max}$, when the two decoders completely agree, i.e., when

$$u_{i,1}^n = u_{i,2}^n, \quad \forall i, 1 \leq i \leq K \tag{1}$$

where $u_{i,1}^n$ and $u_{i,2}^n$ are the ith decoded bits from the first and second decoders, respectively, at iteration n, and K is the information block size. Therefore, agreement on decoded sequences half an iteration apart will cause the iterative process to terminate and the current decoded sequence (from either one of the decoders) to be sent to the output of the turbo decoder. This rule has been considered in [5].

2. Hard Rule H_2 (consistency in two successive iterations). With this rule, the iterative process is stopped when the tentative decoded bit sequence at iteration $n \leq N_{\max}$ is exactly the same as the decoded bit sequence at the previous iteration, i.e., when

$$u_{i,2}^n = u_{i,2}^{n-1}, \quad \forall i, 1 \leq i \leq K \tag{2}$$

We expect that on, the average, this stopping rule should take half an iteration longer than rule H_1. This rule has been considered in [2], along with a soft rule based on cross-entropy that is related to our soft rule S_1 in Section II.B. Cross-entropy stopping rules were originally proposed in [9].

3. Hard Rule H_3 (consistency in three successive iterations). This rule is satisfied at iteration $n \leq N_{\max}$ when the tentative decoded bit sequence at the nth iteration is exactly the same as the previous decoded sequence *and* the decoded sequence two iterations earlier, i.e., if

$$u_{i,2}^n = u_{i,2}^{n-1} = u_{i,2}^{n-2}, \quad \forall i, 1 \leq i \leq K \tag{3}$$

On the average, we expect that this rule should take one iteration longer than rule H_2.

4. Hard Rule H_4 (consistency in four successive iterations). This rule adds one more confirmation to the previous rule, such that the rule is satisfied at iteration $n \leq N_{\max}$ if there is no discrepancy among the tentative decoded sequences at iterations n, $n-1$, $n-2$, and $n-3$:

$$u_{i,2}^{n} = u_{i,2}^{n-1} = u_{i,2}^{n-2} = u_{i,2}^{n-3}, \quad \forall i, 1 \leq i \leq K \tag{4}$$

Note that, among all the rules, this stopping rule is the hardest to meet. Moreover, we expect a smaller probability of undetected errors, at the expense of one or two additional iterations on the average, as compared with rules H_3 and H_2, respectively.

B. Soft-Decision Rules

These are stopping rules based on comparing a metric on bit reliabilities (soft bit decisions) with a threshold. Two simple metrics that can be used to decide when to stop the iterative decoding process are the average and minimum bit reliabilities. At each iteration, the turbo decoder computes a metric on the log-likelihoods (reliabilities) of the information bits and compares it with a preset threshold value. If the metric is smaller than the threshold, the decoder continues with a new iteration; otherwise, it stops.

Let $\lambda_{i,1}^{n}$ denote the ith bit reliability from the first decoder and $\lambda_{i,2}^{n}$ denote the ith bit reliability from the second decoder at iteration n. Then, based on the average and minimum bit reliabilities, we can devise several soft-decision stopping rules.

1. Soft Rule S_1 (full-iteration average bit-reliability check). The average of the absolute values of the bit reliabilities from one of the decoders is compared with a threshold, θ_1. The rule is satisfied at the earliest iteration $n \leq N_{\max}$ such that

$$\frac{1}{K} \sum_{i=1}^{K} |\lambda_{i,2}^{n}| \geq \theta_1 \tag{5}$$

2. Soft Rule S_2 (full-iteration minimum bit-reliability check). The minimum on the absolute value of the bit reliabilities from one of the decoders is compared with a threshold, θ_2. The rule is satisfied at the earliest iteration $n \leq N_{\max}$ such that

$$\min_{0<i\leq K} \left[|\lambda_{i,2}^{n}| \right] \geq \theta_2 \tag{6}$$

3. Soft Rule S_3 (successive half-iteration average bit-reliability check). The minimum on the absolute value of the average reliabilities from both component decoders is compared with a threshold, θ_3. The rule is satisfied at the earliest iteration $n \leq N_{\max}$ such that

$$\min_{0<i\leq K} \left[\frac{1}{2} |\lambda_{i,1}^{n} + \lambda_{i,2}^{n}| \right] \geq \theta_3 \tag{7}$$

4. Soft Rule S_4 (successive half-iteration minimum bit-reliability check). The minimum on the absolute value of bit reliabilities from both decoders is compared with a threshold, θ_4. The rule is satisfied at the earliest iteration $n \leq N_{\max}$ such that

$$\min_{0<i\leq K} \left[|\lambda_{i,1}^{n}|, |\lambda_{i,2}^{n}| \right] \geq \theta_4 \tag{8}$$

5. Soft Rule S_5 (combination of S_3 and S_4). The minimum on the absolute value of bit reliabilities from both decoders and their averages is compared with a threshold, θ_5. The rule is satisfied at the earliest iteration $n \leq N_{\max}$ such that

$$\min_{0<i\leq K} \left[|\lambda_{i,1}^n|, |\lambda_{i,2}^n|, \frac{1}{2}|\lambda_{i,1}^n + \lambda_{i,2}^n| \right] \geq \theta_5 \tag{9}$$

6. Soft Rule S_6 (successive half-iteration bit-by-bit reliability comparison). With this rule, the iterative process is stopped when, for each decoded bit, the soft likelihoods computed by both component decoders are exactly the same, i.e., at the earliest iteration $n \leq N_{\max}$ for which

$$\lambda_{i,1}^n = \lambda_{i,2}^n, \quad \forall i, 1 \leq i \leq K \tag{10}$$

Equality can be satisfied in practice because of finite precision arithmetic.

C. CRC Rule

This is a stopping rule based on detecting erroneous decoded sequences using an outer cyclic redundancy check (CRC) code applied to hard-decoded bits. A separate error-detection code, such as a CRC code, can be concatenated as an outer code with an inner turbo code in order to flag erroneous decoded sequences. The condition for stopping with this rule is satisfied whenever the syndrome of the CRC code is zero. CRC rules were briefly examined in [3,4] without notice made of some of their drawbacks, as discussed later in Section III.B.

The CRC code used for the CCSDS standard [8], with redundancy $\ell = 16$, has a minimum distance of at least 4 for all the turbo frame sizes recommended by the CCSDS and detects at least 99.9985 percent of all frame errors.

D. Magic Genie Rule

In addition to all of the realizable stopping rules defined above, we also define an unrealizable "magic genie" rule, which is useful for establishing an unbeatable performance benchmark against which the other rules are measured. For this rule, the magic genie immediately recognizes the correct decoded word, based on foreknowledge of the transmitted bit sequence, and stops the iterative process in exactly the minimum number of iterations required to produce the correct codeword.

E. Finite Termination Condition for All Rules

All of the above stopping rules (including the magic genie) incorporate an adjunct stopping condition that causes decoding to cease after a maximum of $N_{\max}$ iterations. This prevents an endless loop if the stopping rule is never satisfied.

If the maximum number of iterations is reached before the stopping rule is satisfied, one may use this condition to flag detected errors. Present-day turbo decoders are complete decoders in the sense that they always produce a decoded sequence. Currently, these decoders do not detect and mark unreliable sequences. The stopping rules described here provide a method for accomplishing a degree of error detection as well as a means for increasing the average decoding speed.

In the case of the CRC stopping rule, such error detections are definitive, i.e., a sequence of bits decoded by default at the end of $N_{\max}$ iterations but failing the CRC check cannot possibly be a codeword, but undetected errors are still possible. In the case of the other stopping rules, both undetected and falsely detected errors may occur.

III. Decoder Performance with Stopping Rules

In this section, we present performance results for all of the stopping rules described in Section II. We simulated turbo decoder performance using these rules for turbo codes of rate 1/3 and standard block size 1784, with $N_{\max} = 20$. For the soft-decision rules, we used three different threshold values: a low value, θ^-; a medium value, θ^o; and a high value, θ^+. These values are summarized in Table 1. These threshold values were chosen by trial and error, with the objective of covering a reasonable range of undetected frame-error rates for each rule.

Table 1. Threshold values for soft-decision rules.

Rule	θ^-	θ^o	θ^+
S_1	61.04	76.29	91.55
S_2	7.25	10.30	13.35
S_3	4.20	5.72	7.25
S_4	4.20	5.72	7.25
S_5	3.05	4.20	5.72

Figure 2 shows an example of how the bit reliabilities evolve with iterations. In this figure, bit-reliability traces are shown for every 20th bit in one selected 1784-bit frame at the end of each full iteration. The signal-to-noise ratio for this decoded frame was $E_b/N_0 = 0.6$ dB. This particular frame was decoded successfully, but it required more than the average number of iterations to pass the various stopping conditions. We see that, for this particular frame, all of the bit-reliability traces stay near zero for more than five iterations and then rise sharply to reach limiting values for each bit. For this frame, not one of the bit reliabilities changed after iteration 14. There is a cluster of limiting bit reliabilities with high magnitudes and a few outliers with much lower magnitudes. These outliers with small bit reliabilities are indicative of more serious problems in frames that are not decodable. The soft-decision thresholds should be chosen to guarantee that all of the bit-reliability traces have had a chance to start heading toward (not necessarily reaching) their final values.

The performance with stopping rules is compared with the performance of the magic genie rule and the performance of the standard turbo decoder operating with a fixed number of iterations, $N = 10$. For these performance evaluations, the cost of any required side information (e.g., the CRC code rate or the price of the mythical genie) is neglected. Our criteria for performance comparison of the different stopping rules are decoding speed (average number of iterations per decoded frame), error performance, and computational complexity of the stopping rule.

A. Average Number of Iterations

Figures 3 through 6 show the simulation results for the average number of iterations per decoded frame as a function of the bit signal-to-noise ratio, E_b/N_0, for the CRC, hard-decision, and soft-decision rules with low and high thresholds, θ^- and θ^+, respectively. From these plots, we conclude that, for most of the rules, the average number of iterations is roughly between 4 and 7 for E_b/N_0 near the "waterfall" threshold (the region where the bit-error rate (BER) versus E_b/N_0 code performance changes most abruptly) as compared with the currently used $N = 10$ fixed iterations for the standard turbo decoder. As expected, the average decoding speed increases with E_b/N_0, since decoded sequences converge to the correct codeword in fewer iterations.

Among all of the tested rules, the CRC rule is the fastest on the average, achieving virtually the same decoding speed as the magic genie rule. Both of these rules are guaranteed to be satisfied as soon as the

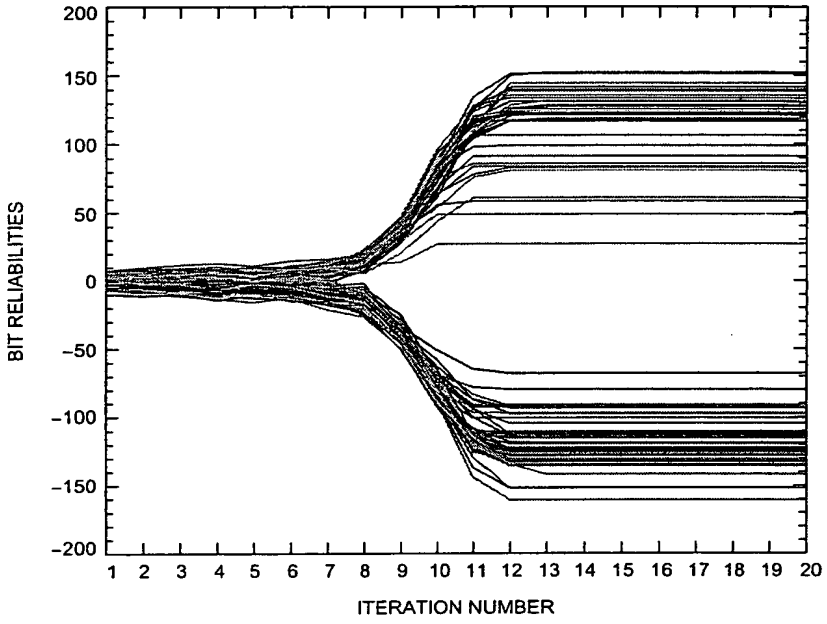

Fig. 2. Evolution of bit reliabilities with iterations for E_b/N_0 = 0.6 dB. Bit reliabilities are plotted for every 20th bit in one selected frame of 1784 bits.

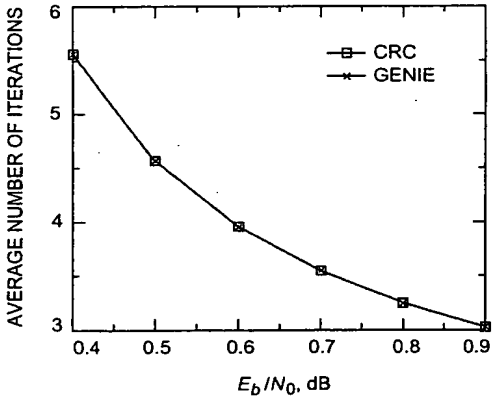

Fig. 3. Average number of iterations for the CRC rule.

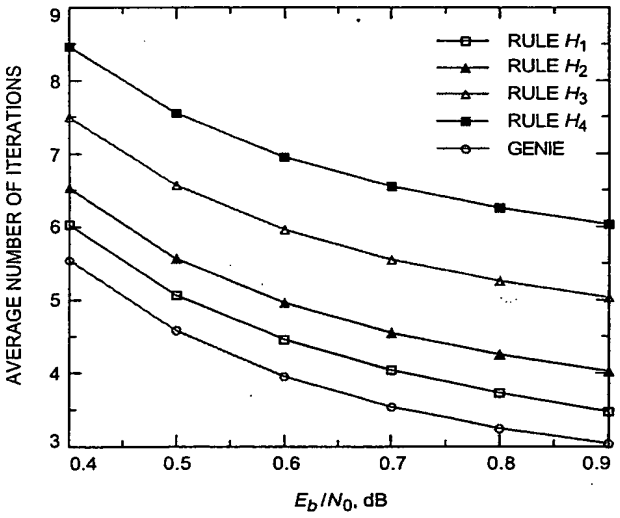

Fig. 4. Average number of iterations for hard-decision rules, H_j.

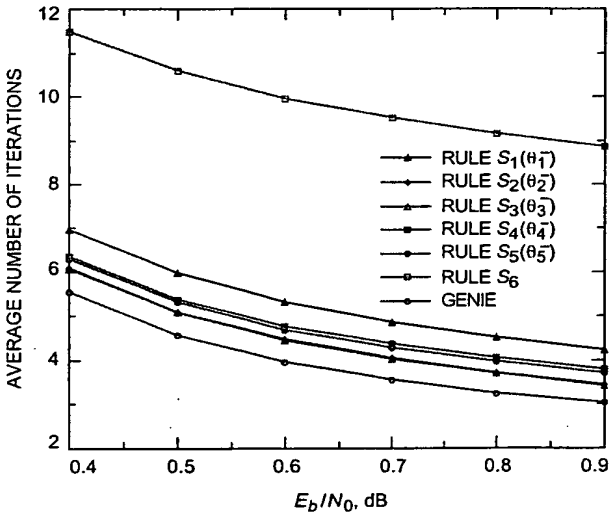

Fig. 5. Average number of iterations for soft-decision rules, $S_l(\theta_l^-)$, with low thresholds, θ_l^-.

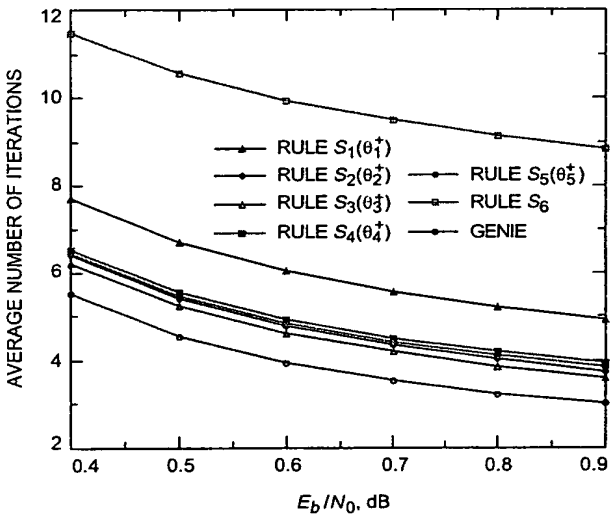

Fig. 6. Average number of iterations for soft-decision rules, $S_l(\theta_l^+)$, with high thresholds, θ_l^+.

true codeword is (tentatively) decoded, and, in fact, the CRC-based decoder outpaces the magic genie decoder by a minuscule amount (on the average) due to occasional premature decoding of false codewords (undetected errors). Figure 4 shows that, on the average, rule H_1 is the fastest of the hard decision rules (one-half iteration slower than the magic genie rule), followed by rule H_2 (one iteration slower than the genie rule), rule H_3 (two iterations slower than the genie rule), and rule H_4 (three iterations slower than the genie rule). For all of these hard-decision rules, the average *extra* number of iterations above that of the genie rule is almost exactly determined by the requisite time to collect the necessary confirmations.

From Figs. 5 and 6, we see that the soft-decision rules also tend to perform within roughly a constant offset from the average number of iterations required by the genie rule. By far, the slowest of the soft rules is S_6, which requires an exact match of all the bit reliabilities from each component decoder and which typically requires about six more iterations than the genie rule. Of the remaining soft rules, the only one significantly slower than the rest is rule S_1, which tests average reliabilities and requires up to two more iterations than the genie, depending on threshold value. All of the other soft rules, S_2 through S_5, perform within about one iteration of the genie rule, even with the higher threshold. By comparing Figs. 5 and 6, we see that, for soft rules S_1 through S_5, the decoding time typically is increased by only a fraction of an iteration if the higher threshold, θ_i^+, is substituted for the lower threshold, θ_i^-.

B. FER and BER Performance

Figures 7 through 10 show the overall (detected-plus-undetected) frame-error rate (FER) and bit-error rate (BER) performance for the CRC, hard-decision, and soft-decision stopping rules with low and high thresholds, θ^- and θ^+, respectively. These graphs also show reference FER and BER performance curves for $N = 10$ fixed iterations per decoded frame and for the magic genie stopping rule with maximum iterations $N_{\max} = 20$. Another reference performance curve, that of $N = 20$ fixed iterations, is not plotted because it is virtually identical to the performance curve for the magic genie with $N_{\max} = 20$. In other words, the decoder with $N = 20$ fixed iterations almost always gets the same answer as the magic genie decoder with $N_{\max} = 20$, but not as quickly as the genie decoder, as seen in the earlier Figs. 3 through 6. This small difference is due to cases where the fixed-iterations decoder gets to the correct answer and then drifts away from it again before stopping.

The results in Figs. 7 through 10 indicate that the overall error rates achieved by turbo decoders employing stopping rules are significantly better than those for $N = 10$ iterations, and in fact are very nearly equal to the error rates achieved by a standard decoder using $N = 20$ fixed iterations or by the magic genie rule with $N_{\max} = 20$. Exceptions occur at higher values of E_b/N_0, where the stopping rules can introduce their own characteristic error floor, higher than the inherent turbo code floor. We see that the FER performance seems to bottom out above 10^{-5} for the CRC rule, for hard rules H_1 and H_2, and for soft rules S_1 through S_5 with the lower threshold θ_i^-. For hard rules H_3 and H_4, and for soft rule S_6, there is no noticeable FER floor down to the limit detectable by the length of the simulations, i.e.,

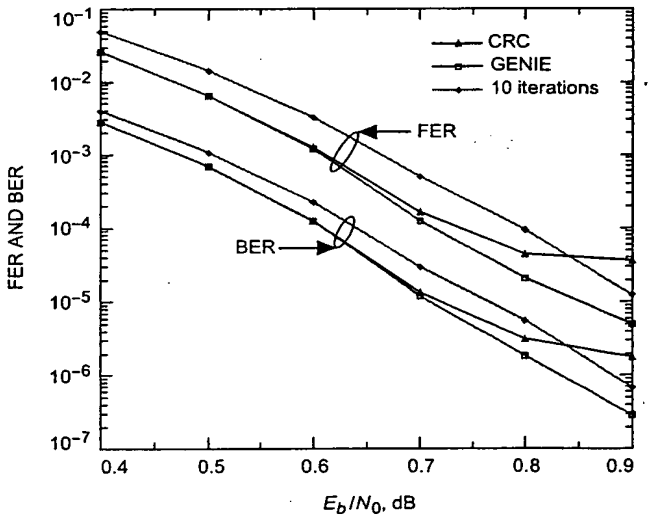

Fig. 7. FER and BER for the CRC rule.

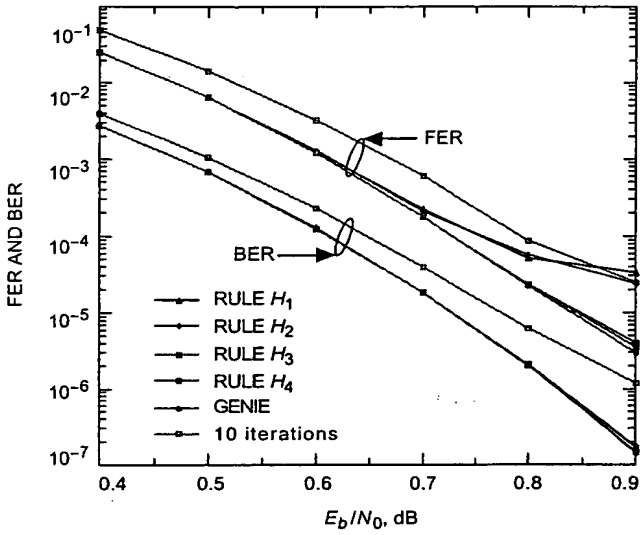

Fig. 8. FER and BER for hard-decision rules, H_l.

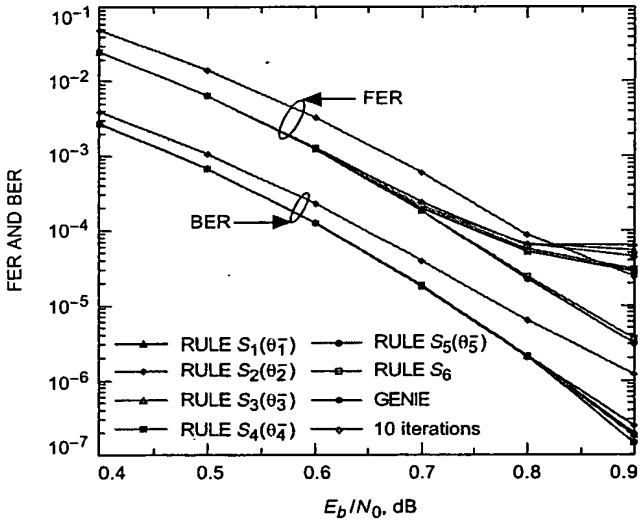

Fig. 9. FER and BER for soft-decision rules, $S_l(\theta_l^-)$, with low thresholds, θ_l^-.

at least to the range between 10^{-5} and 10^{-6}. Also, there are barely noticeable signs of the start of an FER error floor below 10^{-5} for some of the soft rules, S_1 through S_5, with the higher threshold, θ_i^+. In general, the rules that require the least stringent checks are the ones that are most prone to undetected errors and produce the highest characteristic error floors.

The main conclusion from a comparison of Figs. 7 through 10 is that the stopping rules with $N_{\max} = 20$ all do a good job of matching the FER and BER performance of the magic genie with $N_{\max} = 20$ or a standard decoder with $N = 20$ fixed iterations, down to a rule-dependent error floor beyond which

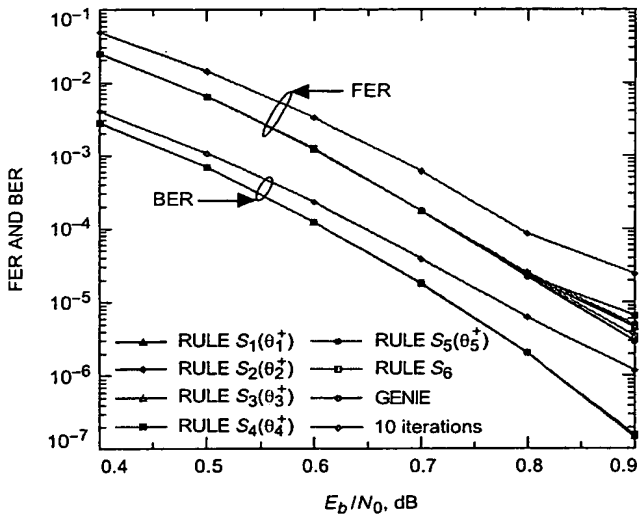

Fig. 10. FER and BER for soft-decision rules, $S_i(\theta_i^\dagger)$, with high thresholds, $\theta_i^\dagger$.

further error-rate improvements are limited by the stopping rule's inability to detect errors 100 percent of the time. Depending on the error-rate requirement, this rule-dependent error floor may provide sufficient cause for rejecting a rule that allows too many erroneous decoded sequences to pass the stopping test. For example, if the FER requirement were below 10^{-5}, then the CRC rule could not be used unless a longer code were substituted (e.g., with $\ell = 32$ instead of $\ell = 16$). Similarly, the easier-to-pass hard-decision rules H_1 and H_2 would have to be rejected in favor of more stringent rules such as H_3 and H_4 that require additional confirmations. The characteristic error floors of the soft-decision rules S_1 through S_5 are adjustable by raising or lowering the corresponding thresholds. This yields greater adaptability of the soft rules to different error-rate requirements, generally with a cost of fewer added iterations, on the average, than the extra confirmations required to lower the error floor for the hard rules.

One of the somewhat surprising, yet obvious in retrospect, findings from the FER and BER performance tests was the poor performance of the 16-bit CRC rule at high SNRs. The FER performance curve in Fig. 7 for this stopping rule seems to bottom out at about 3×10^{-5}, which is about twice the undetected error performance one would expect for this CRC when applied to a totally random received sequence. All of the turbo decoder's iterative efforts are accomplishing nothing toward lowering the unaided undetected error rate of the CRC itself! The explanation for this effect is simple. Because very few codewords are successfully decoded within the first two iterations (see, for example, Fig. 1), virtually all of the codewords will have at least a couple of chances to accidentally pass the CRC check during the early iterations when the tentatively decoded sequences are indeed very random. In this way, the CRC rule can prematurely stop an iterative process that would have eventually produced the correct codeword. Similar premature stopping conditions occur for the other rules, and this leads to their characteristic FER error floors. BER error floors are not noticeable within the range of SNRs tested, except for the CRC rule. When the CRC rule is satisfied prematurely, it usually happens in the earliest iterations, when there is a high percentage of bit errors. In contrast, premature satisfaction of the hard and soft stopping rules typically does not occur unless the bit estimates are correct over most of the frame. For a block size of 1784 bits, we would expect a gap of up to three orders of magnitude between the FER and BER floors, whereas from Fig. 7 this gap is just over one order of magnitude for the CRC rule.

C. Undetected and Falsely Detected Error Rates

Further understanding of the causes and effects of the rule-dependent error floors can be obtained by separately plotting the undetected portion of the overall FER. There are four different conditions that can occur when stopping rules are used by a turbo decoder, depending on whether the decoded sequence is detected to be reliable or unreliable by the decoder and whether it is actually correct or in error. First is the case of correct decoding, when the stopping rule is satisfied at some iteration, $n < N_{\max}$, and the decoded sequence is correct. Second is the case when the stopping rule is satisfied but the decoded sequence is actually in error; this produces an undetected error. Next is the case when the stopping rule fails in $N_{\max}$ iterations, i.e., the decoder detects an unreliable sequence, and the decoded sequence is indeed incorrect; this corresponds to a detected error. Finally, there is the case when the decoder flags a sequence as unreliable, but the decoded sequence is actually correct; this corresponds to a falsely detected error. A good stopping rule has a small undetected error probability and a small probability of falsely detected errors.

Figures 11 through 14 plot the overall FER and the undetected FER for the CRC rule, the hard-decision rules, and the soft-decision rules with low and high thresholds, θ^- and θ^+, respectively. For all of these rules, the undetected error rates stay approximately constant, or decrease very slowly, over the entire range of E_b/N_0 values tested. Also, we observe that at low SNRs the detected errors dominate the overall FER, whereas at high SNRs the undetected errors are more frequent than the detected ones, causing the artificial error floor described in the previous section. In other words, with increasing SNR, the turbo decoder is able to correct more and more errors, but the undetected error rate is relatively impervious to increased SNR and eventually dominates.

In Fig. 15, we analyze the cost in decoding speed for trying to adjust the characteristic error floors of the soft-decision rules S_2 through S_5 by raising or lowering the corresponding thresholds. In this figure, the average number of iterations is plotted as a function of the undetected frame-error rate for several values of E_b/N_0. Each line corresponds to one rule with threshold values θ^+, θ^o, and θ^-, reading left to right. Rule S_1 is not shown because its relatively poorer performance does not keep the curves for each value of E_b/N_0 clustered together. This figure illustrates a general property of the soft rules—that higher threshold values can yield drastically lower undetected frame-error rates with only a small increase in

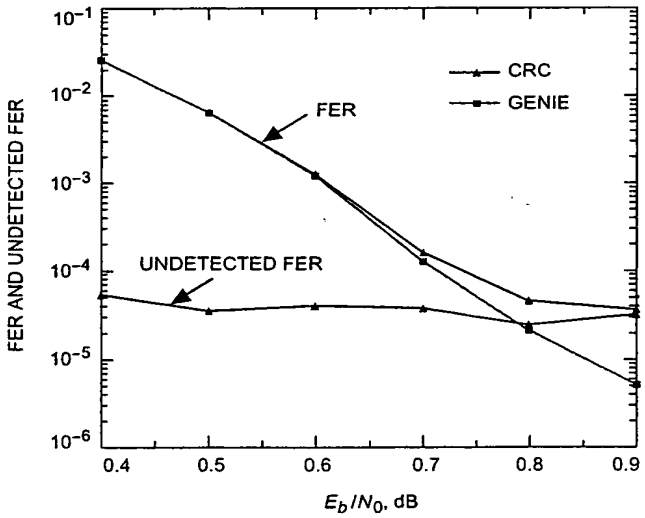

Fig. 11. FER and undetected FER for the CRC rule.

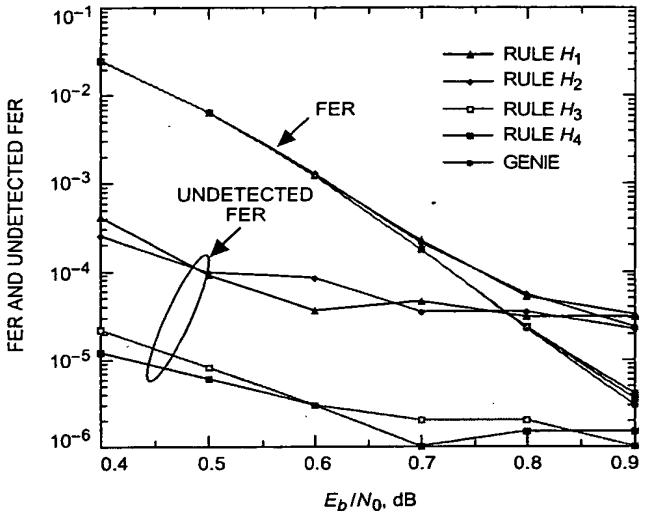

Fig. 12. FER and undetected FER for hard-decision rules, H_i.

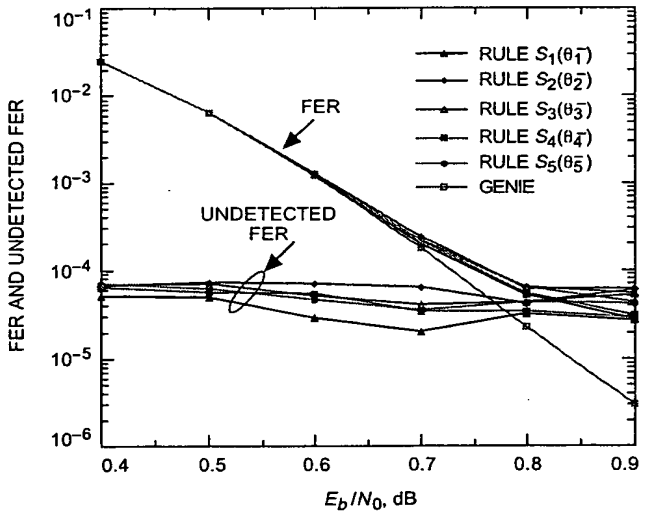

Fig. 13. FER and undetected FER for soft-decision rules, $S_i(\theta_i^-)$, with low thresholds, θ_i^-.

the average number of iterations. The figure also separates the various soft rules according to a distinct preferential ordering: at all of the tested values of E_b/N_0, the soft rule S_3 reaches (with appropriate choice of threshold) a given tolerable undetected FER with fewer iterations (on the average) than the other three rules. This rule first averages the bit reliabilities from the two component decoders, then computes the minimum (absolute) average reliability over all of the bits. The second most efficient soft rule at the tested values of E_b/N_0 is S_2, which requires that all the bit reliabilities from one of the component decoders exceed a minimum absolute value.

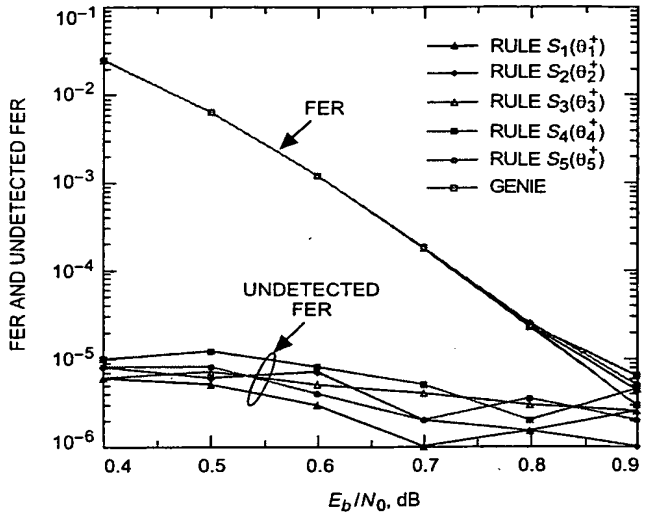

Fig. 14. FER and undetected FER for soft-decision rules, $S_i(\theta_i^+)$, with high thresholds, θ_i^+.

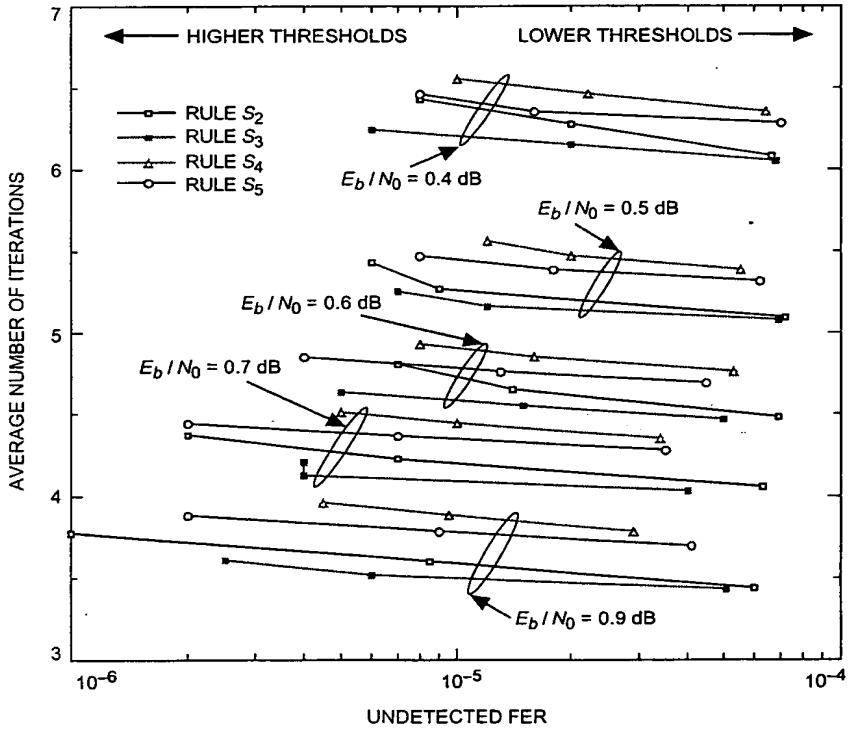

Fig. 15. Effects of threshold selection on decoding speed and undetected FER for soft-decision rules.

Finally, we show in Figs. 16 through 18 the falsely detected error rates for the hard-decision rules and the soft-decision rules with low and high thresholds, respectively. The CRC rule is not plotted, since it has a zero probability of false detection. The falsely detected error rates for all of the other rules fall off steadily with increasing SNR and are generally around an order of magnitude lower than the overall FER for most of the rules. Exceptions are the soft rules S_6 and S_2 at the higher threshold given in Table 1. For the latter rule, the falsely detected FER can be reduced by lowering the threshold. Whether such falsely detected error rates will cause any trouble will depend on the application.

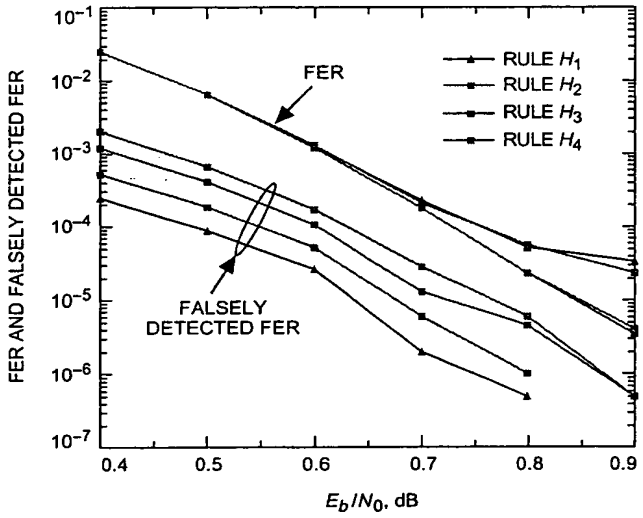

Fig. 16. Falsely detected FER for hard-decision rules, H_i.

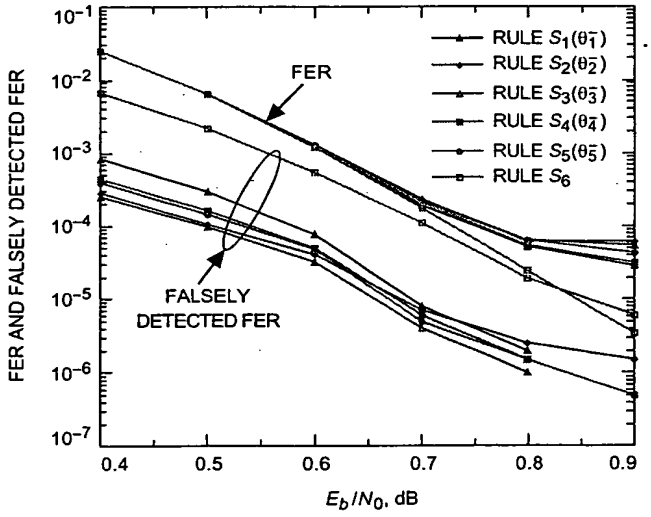

Fig. 17. Falsely detected FER for soft-decision rules, $S_l(\theta_l^-)$, with low thresholds, θ_l^-.

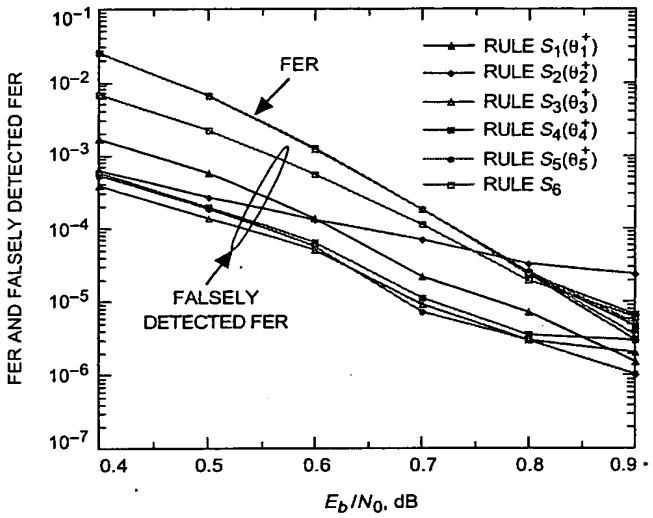

Fig. 18. Falsely detected FER for soft-decision rules, $S_i(\theta_i^+)$, with high thresholds, θ_i^+.

D. Decoding Speed and Error-Rate Performance Trade-Offs

In order to compare the stopping rules, we examine the overall frame-error rate performance as a function of the average decoding speed normalized to the genie speed for all the simulated SNR values. The results are shown in Fig. 19, where we also show the performance of turbo decoders with $N = 10$, 15, and 20 fixed number of iterations. The genie rule is depicted for reference at 100 percent decoding speed, which translates into only 3 to 6 iterations necessary on average to achieve virtually the same performance as the decoder with 20 iterations. The rest of the rules are depicted as individual points on the graph for the hard-decision rules H_1 through H_4, or as connected points for three different threshold values (high, medium, and low) in the case of the soft-decision rules S_1 through S_5. Also shown for reference is the absolute speed (measured in average number of iterations) for the genie as a function of FER.

There are several interesting observations that we can draw from the results in Fig. 19:

(1) The average number of iterations required by the genie ranges from about 6 iterations at higher FERs down to about 3 iterations at lower FERs. Note that in Fig. 19 the value of E_b/N_0 is changed appropriately to achieve the desired FER, whereas in Fig. 1 the value of E_b/N_0 was held fixed. This explains the paradoxical result that FER decreases with *increasing* iterations in Fig. 1 at fixed E_b/N_0, whereas it takes a *decreasing* average number of iterations to reach decreasing FER levels at the expense of appropriately higher E_b/N_0 in Fig. 19.

(2) Decoding using a fixed number of iterations achieves only 30 to 60 percent of genie speed with $N = 10$ iterations and 15 to 30 percent of genie speed with $N = 20$ iterations.

(3) Decoding using a fixed number of iterations exacts a relatively steep price in terms of higher frame-error rates in return for increasing the decoding speed. This is indicated by the moderately high slopes of the lines at different values of E_b/N_0 for the fixed-iterations rules.

(4) Decoding using any of the hard or soft stopping rules achieves FER performance very close to that of $N = 20$ fixed iterations, down to frame-error rates as low as about 10^{-4}.

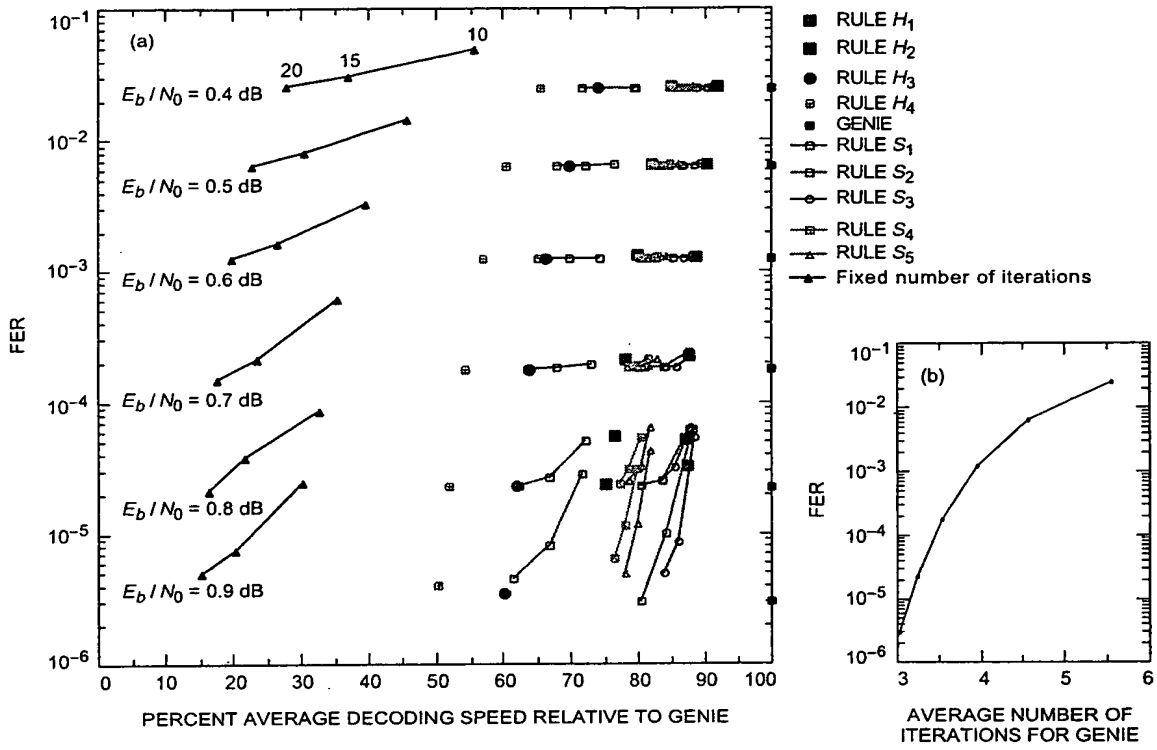

Fig. 19. Comparison of overall frame-error rate performance: (a) performance versus average decoding speed relative to the Genie and (b) absolute speed of the Genie.

(5) Decoding speed using the best stopping rules can be pushed toward 90 percent of genie speed with only a relatively small increase in error rates, down to frame-error rates of about 10^{-4}. This is indicated by the very shallow upward trends of the symbols corresponding to the variable-iterations stopping rules for values of E_b/N_0 from 0.4 to 0.7 dB.

(6) For frame-error rates around 10^{-5} or lower, it becomes very difficult to push the speed of the stopping-rule decoders toward 90 percent of genie speed without paying dearly in error-rate performance. This is indicated by the steep upward trend of the symbols corresponding to the variable-iterations stopping rules for values of E_b/N_0 from 0.8 to 0.9 dB. However, decoding using the best stopping rules (with soft-decision rules and high threshold values) can still achieve decoding speeds of 75 percent or more of genie speed, while also maintaining FER performance in the 10^{-5} to 10^{-6} range very close to that of $N = 20$ fixed iterations. This represents an approximate 5-fold average speed improvement with essentially the same performance as compared with a decoder with $N = 20$ fixed iterations and an approximate 2.5-fold speed improvement together with an order-of-magnitude FER improvement as compared with a decoder with $N = 10$ fixed iterations.

(7) The most efficient of the soft rules again appears to be S_3, followed by S_2, although the difference is only noticeable for required frame-error rates below 10^{-4}. These are the same conclusions drawn from Fig. 15 based on analyzing the average number of iterations

versus undetected errors only. Rule S_1, which was not plotted in Fig. 15, emerges as the clear loser among the soft rules. This rule was actually the first one we tested, as it is based on the simple notion that the decoder iterations should stop when the (absolute) bit reliabilities averaged over all the bits exceeds a threshold. However, the rules that check for a *minimum value* over all bits rather than an *average value* are clearly more efficient.

(8) Among the hard-decision rules, rule H_1, which performs consistency checks each half-iteration, is clearly faster and performs nearly as well as rule H_2, which only checks from one full iteration to the next. Rule H_3, which requires a double confirmation, clearly outperforms rules H_1 and H_2 for frame-error rate requirements below 10^{-4}, but it pays a significant penalty in speed. A soft rule such as S_2 or S_3 using a high threshold can achieve the same lower error rate while requiring significantly fewer iterations on the average. Rule H_4, requiring a triple confirmation, does not give any noticeable performance improvement versus rule H_3, and thus the extra iteration required for its extra confirmation is essentially wasted.

The main advantage of using stopping rules is a considerable increase in decoding speed. On the other hand, the overall error rate is increased if the stopping condition is satisfied prematurely. However, with soft-decision rules, we can essentially eliminate these undetected errors by properly adjusting the threshold value. Another good engineering solution is available by concatenating an outer CRC code with a turbo code employing one of the hard or soft stopping rules. When the CRC code is only used as an error detector once the iterations have been stopped by a different rule, we expect to obtain an undetected error probability about 3×10^{-5} times lower than the frame-error rate of the turbo code alone.

E. Buffering Requirements Due to Variable Decoding Speed

Some of the substantial performance and/or speed advantages of a decoder using stopping rules may be offset by the need for increased buffering of data to accommodate the variability in the number of iterations required to decode each frame. For example, suppose that the decoder using stopping rules can decode frames with an average of 4 iterations, and that the designers of the communication system take full advantage of this by constantly feeding the decoder with a new input frame after every fourth iteration. If the decoder should encounter a few successive frames that require the maximum $N_{\max} = 20$ iterations, it will build up a backlog of many frames. Of course, a frame requiring $N_{\max} = 20$ iterations occurs only with low probability, given by the sum of the falsely detected FER and the overall FER minus the undetected FER, as can be calculated from Figs. 16 through 18 and Figs. 11 through 14. More likely is the event that a string of consecutive frames each requires a few more iterations than the long-term average, and this also leads to a backlog of frames waiting to be decoded. In either case, an appropriately sized buffer is needed to accommodate the backlogged frames or else some of these frames will be lost.

The amount of extra buffering required by a decoder using stopping rules is determined primarily by two factors: the statistical frame-to-frame variability of the actual number of decoder iterations and the variability of the input frame rate to the decoder. The first factor is illustrated by the scenario described in the previous paragraph. The second factor does not ameliorate the total buffering requirements, but it may lower the amount of buffering "chargeable" to the variable-speed decoder. In other words, if a certain amount of buffering is already needed to accommodate a variable-input frame rate, the variable-speed decoder may add very little to the existing requirement.

We do not undertake a detailed analysis of these effects here, but provide only a brief outline of an analytical method for future work. Suppose that the ith input frame arrives at the decoder after the decoder has had time to perform m_i iterations since the arrival of the $(i-1)$th input frame. After the arrival of I such input frames, the decoder will have had time to perform a total of $N_{tot} = \sum_{i=1}^{I} m_i$ iterations. If n_j iterations are required to decode the jth output frame, then during this time the decoder

will have had time to decode J output frames, where J is the largest integer satisfying $\sum_{j=1}^{J} n_j \leq N_{tot}$. If the buffer can hold a total of B frames, then frames will be lost any time the accumulated number of input frames exceeds the accumulated number of decoded frames by more than B, i.e., $I - J > B$. To fully model this problem, one must also account for the possibility that the decoder may be idle during periods of infrequent arrivals.

If the communication system is designed to take full advantage of the average throughput capability of the stopping-rule decoder, the average number of iterations between frame arrivals, $E(m_i)$, will equal the average number of decoder iterations, $E(n_j)$. Then the probability that a frame is lost to buffer overflow will depend on the variability around these mean values of both the arrival sequence, $\{m_i\}$, and the decoding sequence, $\{n_j\}$. A typical distribution for the required number of decoder iterations is shown in Fig. 20, where we plot $\Pr[n_j > N]$ for the hard-decision rules H_1 through H_4 at $E_b/N_0 = 0.6$ dB.

An improved buffering scheme, which will be the subject of future work, can be designed by identifying the speed of convergence for each frame in the initial iterations and then giving priority to the frames that promise to decode faster. This method will achieve a lower overflow probability for a given total buffer size, but it will require more complex buffering and an additional method for abandoning and properly resuming the decoding of a frame by saving the decoder internal state.

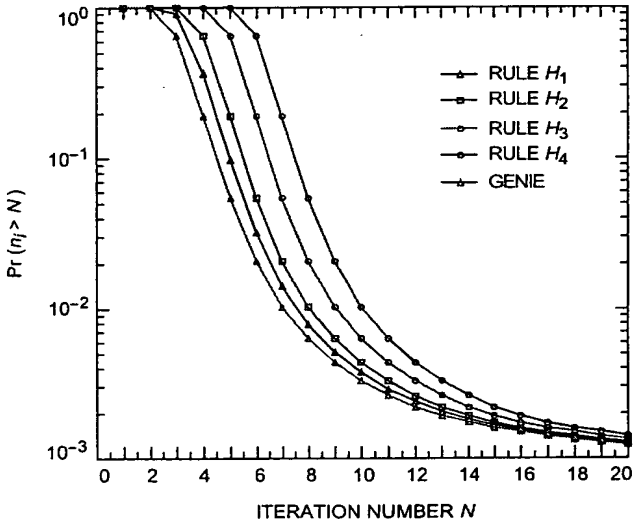

Fig. 20. Complementary cumulative distribution function for the number of iterations, for hard-decision rules, H_l.

F. Computational Complexity

A fixed-iterations turbo decoder passes only extrinsic information during its iteration cycle, and it needs only to calculate bit reliabilities at the end of the final iteration as a prelude to its final bit decisions. In contrast, the soft-decision stopping rules, as defined here, require bit-reliability computations at each iteration or half-iteration. This represents extra computations to add the prior likelihoods to the extrinsics to obtain the bit reliabilities, unless the decoder already is obtaining its extrinsics from the bit reliabilities by subtracting the prior likelihoods. If the decoder normally bypasses explicit computation of the bit reliabilities until the final iteration, it may be computationally more efficient to apply rules S_1 through S_6 to the set of extrinsics, rather than the set of reliabilities. The hard-decision rules require tentative bit decisions, based on the computed bit reliabilities, at each iteration or half-iteration. These bit decisions

are additional computations as compared with those of a standard fixed-iterations turbo decoder, which needs to make bit decisions only at the end of the final iteration. It should be emphasized that the extra computations to obtain the bit reliabilities and/or the tentative bit decisions must be performed only once *per decoded bit* per iteration or half-iteration, whereas the computations required during the decoder's normal backward–forward decoding procedure occur several times *per trellis edge* per iteration or half-iteration. Thus, for the turbo codes proposed for the CCSDS, for which there are 32 trellis edges per decoded bit per half-iteration, the relative computational complexity of the basic operations required for turbo decoding is around two orders of magnitude higher than the additional operations required to compute some of the simpler stopping rules analyzed here. Somewhat higher computational investments are needed for the more complex rules that require the most logic to test the stopping condition in terms of the computed bit reliabilities or tentative bit decisions (e.g., the CRC rule).

In terms of computational complexity (number of operations per iteration), the turbo decoder stopping rules can be divided into two categories: rules with no memory/storage requirements for soft or hard bit decisions from past iterations and rules that require memory/storage for bit decisions from earlier iterations. Among the rules in the first category, soft rules S_1 and S_2 have the lowest complexity, while the CRC rule is the most computationally demanding. In the second category, hard rules H_1 and H_2 and soft rule S_6 require the fewest operations per iteration, whereas soft rule S_5 has the highest computational complexity.

Some of the computational overhead needed to check for the stopping condition can be eliminated if the testing of the stopping condition is skipped during the first couple of iterations when there is little payoff for applying the test, because few of the codewords are decodable at this early stage. Thus, it may be worthwhile to develop heuristics that govern when a stopping condition should be tested and when it should be skipped.

IV. Conclusion

Our results indicate that effective stopping rules have the ability to increase the average decoding speed while also improving decoder performance. We tested several simple stopping rules, the best of which provide a significant increase in the average decoding speed, from 50 to 75 percent or more of genie speed at low error rates, while maintaining FER performance very close to that of 20 fixed iterations.

Although hard- and soft-decision rules can achieve similar savings in the average number of iterations, soft rules offer more control of the trade-off between undetected and falsely detected errors, assuming that the threshold can be set carefully.

One of the stranger conclusions from the speed-versus-performance trade-off analysis is that averaging the bit reliabilities across the two component decoders makes for a good test statistic (e.g., S_3), whereas further averaging the bit reliabilities over all the bits creates a poorer test statistic (e.g., S_1). The better stopping rules use test statistics that compute minima rather than averages over all the bits.

A second strange conclusion concerns the efficacy of an error-detection code such as a CRC code. We have seen that testing the tentative bit decisions using a 16-bit CRC code is a poor choice for a stopping rule as compared with the other ad hoc hard and soft rules, which are more easily computable and better at eliminating undetected errors. A stopping condition based on *any* error-detecting code will suffer from the same type of problem, namely there will be a floor on the *overall FER* that is several times higher than the *undetected error probability* of the code, unless the test is skipped during the early iterations, in fact until the turbo decoder has a chance to successfully decode the vast majority of input frames so as not to allow random-looking intermediate results to accidentally satisfy the error-detection decoder. Of course, waiting too long for the decoder to successfully decode most of its frames before testing any stopping condition defeats the purpose of using a stopping rule in the first place! However, an error-detecting code can still have an effective role in a system using stopping rules if it is tested only once as part of a system

that uses one of the other stopping rules to first diminish the likelihood that an incorrect frame will be fed to the error-detecting decoder. In this case, the low overall FER achieved by the stopping rule will be lowered additionally by a factor roughly equal to the undetected error probability of the error-detecting code. Thus, a 16-bit CRC code, checked once after a decoding process using a stopping rule such as S_3 that is capable by itself of lowering the undetected FER below 10^{-5}, can further lower the undetected FER below 10^{-9}. Stronger CRC codes could be used to lower the undetected FER to any desired level, at a computational expense that does not depend on the number of turbo decoder iterations.

Further work is needed to quantify the buffering requirements and the computational complexity required to test the stopping conditions. Also, it should be emphasized that the results obtained in this article are all based on one turbo code, with rate 1/3 and block size 1784, and it is important to determine whether similar conclusions are obtained for different turbo and turbo-like codes. Another fruitful area of future research is to obtain a theoretical understanding of how the numerical threshold values for the soft-decision rules should be chosen to give the best trade-off between average decoding speed and overall FER or undetected FER, and also how these stopping thresholds on minimum bit reliabilities are related to other ad hoc thresholds often used in practical turbo decoder implementations to clip the computed values of the extrinsics in order to avoid numerical problems.

The selection of $N_{\max} = 20$ was arbitrary, and it would be interesting to determine the ultimate performance improvement of stopping rules when a much higher maximum limit is substituted (e.g., $N_{\max} = 200$). Such an increase in $N_{\max}$ would have only a very small impact on the *average* decoding speed, due to the very small number of codewords that would ever require more than 20 iterations at values of E_b/N_0 above the normal decoding threshold.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding: Turbo Codes," *Proc. 1993 IEEE International Conference on Communications*, Geneva, Switzerland, pp. 1064–1070, May 1993.

[2] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two Simple Stopping Criteria for Turbo Decoding," *IEEE Transactions on Communications*, vol. 47, pp. 1117–1120, 1999.

[3] A. Shibutani, H. Suda, and F. Adachi, "Reducing Average Number of Turbo Decoding Iterations," *Electronics Letters*, vol. 35, pp. 701–702, 1999.

[4] A. Shibutani, H. Suda, and F. Adachi, "Complexity Reduction of Turbo Decoding," *Vehicular Technology Conference, VTC 1999*, vol. 3, pp. 1570–1574, 1999.

[5] K. Gracie, S. Crozier, and A. Hunt, "Performance of a Low-Complexity Turbo Decoder with a Simple Early Stopping Criterion Implemented on a SHARC Processor," *IMSC 99, Proceedings of the Sixth International Mobile Satellite Conference*, Ottawa, Ontario, Canada, pp. 281–286, 1999.

[6] B. Kim and H. S. Lee, "Reduction of the Number of Iterations in Turbo Decoding Using Extrinsic Information," *Proceedings of IEEE TENCON 99*, Inchon, South Korea, pp. 494–497, 1999.

[7] Consultative Committee for Space Data Systems, "Telemetry Channel Coding," Blue Book, vol. 101.0-B-4, issue 4, May 1999.
http://www.ccsds.org/documents/pdf/CCSDS-101.0-B-4.pdf

[8] Consultative Committee for Space Data Systems, "Packet Telemetry," Blue Book, vol. 102.0-B-2, January 1987 or later issue.

[9] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, March 1996.